SAMSUNG
Open Source Group

Tutorial
Prototyping IoT devices on GNU/Linux

Embedded Linux Conference
#LFELC, Berlin, Germany <2016-10-13>

Philippe Coval
Samsung Open Source Group / SRUK
philippe.coval@osg.samsung.com

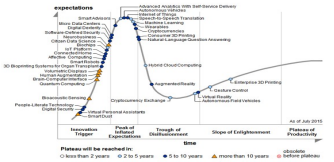
Hallo Welt!

SAMSUNG
Open Source Group

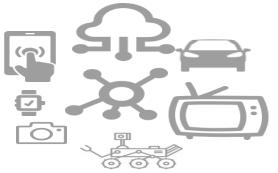
- Philippe Coval
  - Software engineer for Samsung OSG
    - Belongs to SRUK team, based in Rennes, France
    - Ask me for IoTivity support on Tizen platform and others
  - Interests
    - Libre Soft/Hard/ware, Communities, Interoperability
      - DIY, Embedded, Mobile, Wearables, Automotive...
  - Find me online
    - https://wiki.tizen.org/wiki/User:Pcoval

Newbies, makers, hackers welcome !

SAMSUNG
Open Source Group



- This “IoT” talk is not about:
  - Market share, prospects, growth, figures
    - Monetize data with cloud, analytics, big data, machine learning
      - Security, privacy, trust, Skynet, singularity or any concerns
  - Architectures, services or designs
    - Comparison of protocols or implementations
    - Tizen the “OS of Everything” (unless if asked)



- It's about **quick** **prototyping** for proof of concepts:
  - Learn by doing from scratch, DIY: software, hardware, electronics
  - Feedback on previous **experimentations** from embedded developer
  - Front door to a project of 435K+ lines of code and ~500 pages of specifications

Agenda

SAMSUNG
Open Source Group

- Prototyping
- Simplest example
- Implementation
- Hardware integration
- Demonstration
- Q&A

Motivations for prototyping

SAMSUNG
Open Source Group

- *NOT* making a mass produced IoT device at 1st shot
  - Low cost (<10 $), low consumption (mW), high level of security
- Validate concepts with **relaxed constraints**
  - In **friendly environment** (ie: tools, security or connectivity shortcuts)
  - Validate, show, gather feedback, stress, benchmark, adapt, iterate
- Think of use cases first?
  - Or experiment with what can technology can provide? Be inspired!
- Topics and Ideas?
  - Controlling, monitoring, convergence, network of sensors, behaviors, AI...

SAMSUNG
Open Source Group

"**Simplicity**
is the ultimate sophistication."
~Leonardo da Vinci

Simplest use case

SAMSUNG
Open Source Group

- From the blinking led
- To a remote controlled switch
  - GPIO, LED, Relay, Motor, Fan, Home Appliance...
  - Simple functions: On/Off
- To a flip/flop relay controlled by multiple clients
  - Notification of change in real time
  - Consistent toggle feature
- Identified problems, are half solved :
  - **Sharing** hardware resource(s) through a seamless **connectivity**

IoTivity : Connectivity between devices

SAMSUNG
Open Source Group

- Apache-2 licensed C/C++ Implementation
  - Of Open Connectivity Foundation's **standard** (OCF~OIC)

- Many features:
  - **Discovery** (IETF RFC7252 / IP Multicast)
  - Communication (RESTfull API on CoAP) w/ Security (DTLS)
  - Transports (IP, WiFi, BT, BLE, Zigbee...)
  - Data/Device management, web services, cloud, plugins...

- Today we'll use only few features to connect our thing

OCF Vocabulary is all about resources

SAMSUNG
Open Source Group

- Resource is representing
  - virtual object (ie: logical states)
  - physical data (ie: actuator, sensors)
  - hybrid (ie: soft sensors)
- Resource entity
  - Each can be accessed by an URI
  - Has a resource type identifier
  - Is composed of **properties**
    - type, name, value

- More concepts
  - Model to describe
    - Resource's **interface**
      - Properties & allowed ops
      - GET, POST, PUT, params...
  - Groups, collections, links
  - Scenes, Things manager
  - Many more services

Don't reinvent the wheel

SAMSUNG
Open Source Group

- OCF's Standardized data **model** repository
  - http://www.oneiota.org/
  - **RESTful** API Modeling Language (RAML > JSON)
  - To be used with a simulator (ATM)
- Search for existing models
  - https://github.com/OpenInterConnect/IoTDataModels
  - http://www.oneiota.org/documents?q=switch
    - binarySwitch.raml includes oic.r.switch.binary.json
    - http://www.oneiota.org/revisions/1580

OCF Model defines switch resource type

SAMSUNG
Open Source Group

```
{ "id": "http://openinterconnect.org/iotdatamodels/schemas/oic.r.switch.binary.json#",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
  "title": "Binary Switch",
  "definitions": {
      "oic.r.switch.binary": {
        "type": "object",
        "properties": {
          "value": {
            "type": "boolean",
            "description": "Status of the switch"
} } } } // ...
```

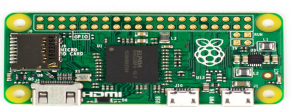
SAMSUNG
Open Source Group

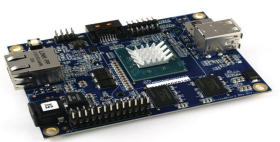
"The secret of getting ahead
is getting **started**."
~ Mark Twain

Time to make choice

SAMSUNG
Open Source Group

- OS? https://wiki.iotivity.org/os
  - None: for Microcomputers (MCU: Bare metal)
  - **GNU/Linux** : Debian/Ubuntu, Yocto, Tizen, OpenWRT...
  - Or others FLOSS or not
- Hardware? https://wiki.iotivity.org/hardware
  - Arduino (MCU) : C API
  - Cheap **Single Board Computer** (CPU): C++ API (or C API too)
    - IO: GPIO, I2C, SPI, Antennas, Daughter-boards...
    - RaspberryPI (0|1|2|3), MinnowMax (OSHW), Edison, Samsung ARTIK (5|10), ...





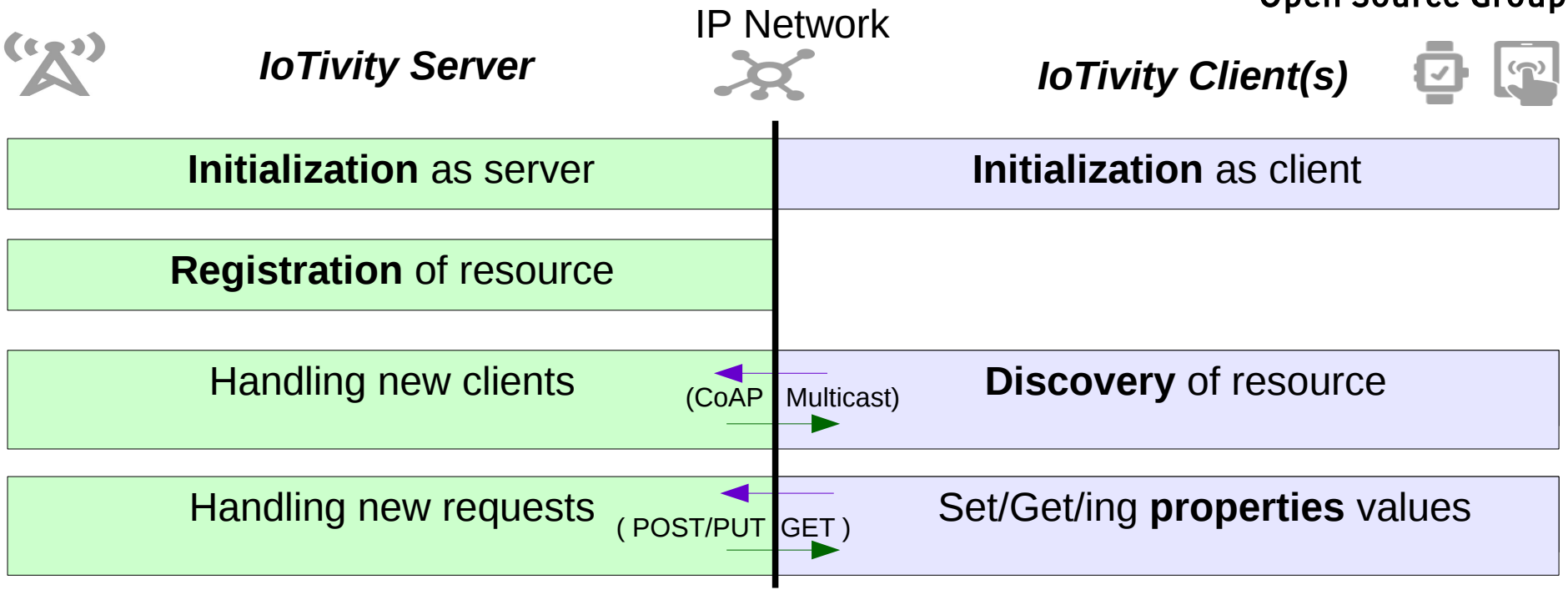
Get your hands on IoTivity!

SAMSUNG
Open Source Group

- Get and build libraries: https://wiki.iotivity.org/build
  - Download sources and dependencies
    - Build it using **scons**
  - Or if OS shipping IoTivity (Tizen, Yocto, ...)
    - Use it a regular **library** (CPPFLAGS & LDFLAGS)
- Look at tree: https://wiki.iotivity.org/sources
  - Samples apps: resource/examples
  - C++ SDK: resource/resource/src
  - C SDK: resource/csdk

Typical flow

SAMSUNG
Open Source Group



- Minimal example project to base on: git clone iotivity-example
  - Simple C (uses callbacks) or C++11

SAMSUNG
Open Source Group

"Talk is cheap.
Show me **the code**."
~ Linus Torvalds

Initialization

SAMSUNG
Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

OCPlatform::Configure(OC::PlatformConfig)	OCPlatform::Configure(OC::PlatformConfig)

```
class IoTServer {
  int main() { init(); … }

  OC::PlatformConfig mPlatformConfig;
  void init() {
    mPlatformConfig =  OC::PlatformConfig
        (OC::ServiceType::InProc,
         OC::ModeType::Server, // different that C
        "0.0.0.0", 0, // default for all subnets / ifaces
         OC::QualityOfService::LowQos //or HighQos
        );
    OCPlatform::Configure(mPlatformConfig);
  }
};
```

```
class IoTClient {
  int main() { init(); … }

  OC::PlatformConfig mPlatformConfig;
  void init() {
    mPlatformConfig = OC::PlatformConfig
        (OC::ServiceType::InProc,
         OC::ModeType::Client, // different than S
         "0.0.0.0", 0, // on any random port available
         OC::QualityOfService::LowQos // or HighQos
         );
    OCPlatform::Configure(mPlatformConfig);
  }
};
```

Registration of resource on Server

SAMSUNG
Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

OCPlatform::Configure(PlatformConfig) **OCPlatform::registerResource(...)**	OCPlatform::Configure(PlatformConfig) OCPlatform::findResource(...)

```
class IoTServer { // (...)
    OCResourceHandle mResource;
    OC::EntityHandler mHandler; // for CRUDN operations
    void setup() { // (...)
        result = OCPlatform::registerResource(mResource, // handle for resource
            "/BinaryRelayURI", // Resource Uri,
            "oic.r.switch.binary", "oic.if.baseline" // Type & Interface (default)
            mHandler // Callback to proceed GET/POST (explained later)
            OC_DISCOVERABLE | OC_OBSERVABLE // resource flags
           );
       OCPlatform::bindTypeToResource(mResource, … ); // optionally
}    };
```

Resource discovery on client : finding

SAMSUNG
Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

OCPlatform::Configure(OC::PlatformConfig) OCPlatform::registerResource(...) { OCPlatform internal }	OCPlatform::Configure(OC::PlatformConfig) **OCPlatform::findResource**(OC::FindCallback) IoTClient::onFind(OCResource)

```
class IoTClient{ // ...
    OC::FindCallback mFindCallback;
    void onFind(shared_ptr<OCResource> resource);
    void setup() { //…
        mFindCallback = bind(&IoTClient::onFind, this, placeholders::_1); //C++11 std::bind
        OCPlatform::findResource("", // default
            "/oic/res", // CoAP endpoint, or resource based filtering for switches
            CT_ADAPTER_IP, // connectivityType can BT, BLE or other supported protocol
            mFindCallback, // to be called on Server response
            OC::QualityOfService::LowQos // or HighQos
        );
 } };
```

Resource discovered on client

SAMSUNG Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

OCPlatform::Configure(OC::PlatformConfig)
OCPlatform::registerResource(...)
{ OCPlatform internal }

OCPlatform::Configure(OC::PlatformConfig)
OCPlatform::findResource(OC::FindCallback)
IoTClient::onFind(OCResource)

```
class Resource { OCResourceHandle mResourceHandle; }; // Our resource for CRUDN

class IoTClient { // (…)
    std::shared_ptr<Resource> mResource;
    void onFind(shared_ptr<OCResource> resource)  {
        if (“/BinarySwitchURI” == resource->uri())
            mResource = make_shared<Resource>(resource);
    }
};
```

Resource discovering on client

SAMSUNG
Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

OCPlatform::registerResource(...) IoTServer::**handleEntity**(OCResourceRequest	OCPlatform::findResource(...) IoTClient::mResource->**post**()

```
void IoTServer::setup() { //...
    OC::EntityHandler handler = bind(&IoTServer::handleEntity, this, placeholders::_1);
    OCPlatform::registerResource( … handler … );  ...
}
IoTServer::handleEntity(shared_ptr<OCResourceRequest> request) {
       string requestType = request->getRequestType();
       if ( requestType == “POST” ) { handlePost() } else { … }
       auto response = std::make_shared<OC::OCResourceResponse>(); //...
       OCPlatform::sendResponse(response);
 }
void IoTServer::handlePost(...) {}
```

Resource representation

SAMSUNG
Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

OCPlatform::registerResource(...)
IoTServer::handleEntity(OCResourceRequest)
IoTServer::handlePost(OCResourceRequest)

OCPlatform::findResource(...)
IoTClient::mResource->**post**(false)

```
void Resource::post(bool value) {
   OCRepresentation rep;  QueryParamsMap params;
   rep.setValue("value", value); // property
   mOCResource->post(rep, params, mPostCallback);
```

```
 IoTServer::handlePost(shared_ptr<OCResourceRequest> request) {
    OCRepresentation requestRep = request->getResourceRepresentation();
    if (requestRep.hasAttribute("value")) {
          bool value = requestRep.getValue<bool>("value");
          cout << "value="<<value<<endl; // OR set physical IO (GPIO...)
}  }
```

GET / POST using Entity Handler

SAMSUNG
Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

```
OCPlatform::Configure(OC::PlatformConfig )
OCPlatform::registerResource(...)
```

```
OCPlatform::Configure(OC::PlatformConfig )
OCPlatform::findResource(...)
```

```
OC::EntityHandler(OCResourceRequest)  {
  switch(getRequestType) {
    case 'POST: // Create resource 1st
      …
    case 'GET' : // Retrieve current value
      ...
    case 'PUT' : // Not allowed for Switch
      ...
   OCPlatform::sendResponse(...);
     OCPlatform::notifyAllObservers();
 }}
```

```
OC::OCResource::post(...) // Create
    OC::PutCallback(...)
```

```
OC:ObserveCallback(...) // Notify
```

```
OC::OCResource::get(...) // Retrieve
    OC::GetCallback(...)
```

SAMSUNG
Open Source Group

"I'm not crazy. My **reality**
is just different from yours."
~ *Lewis Carroll*

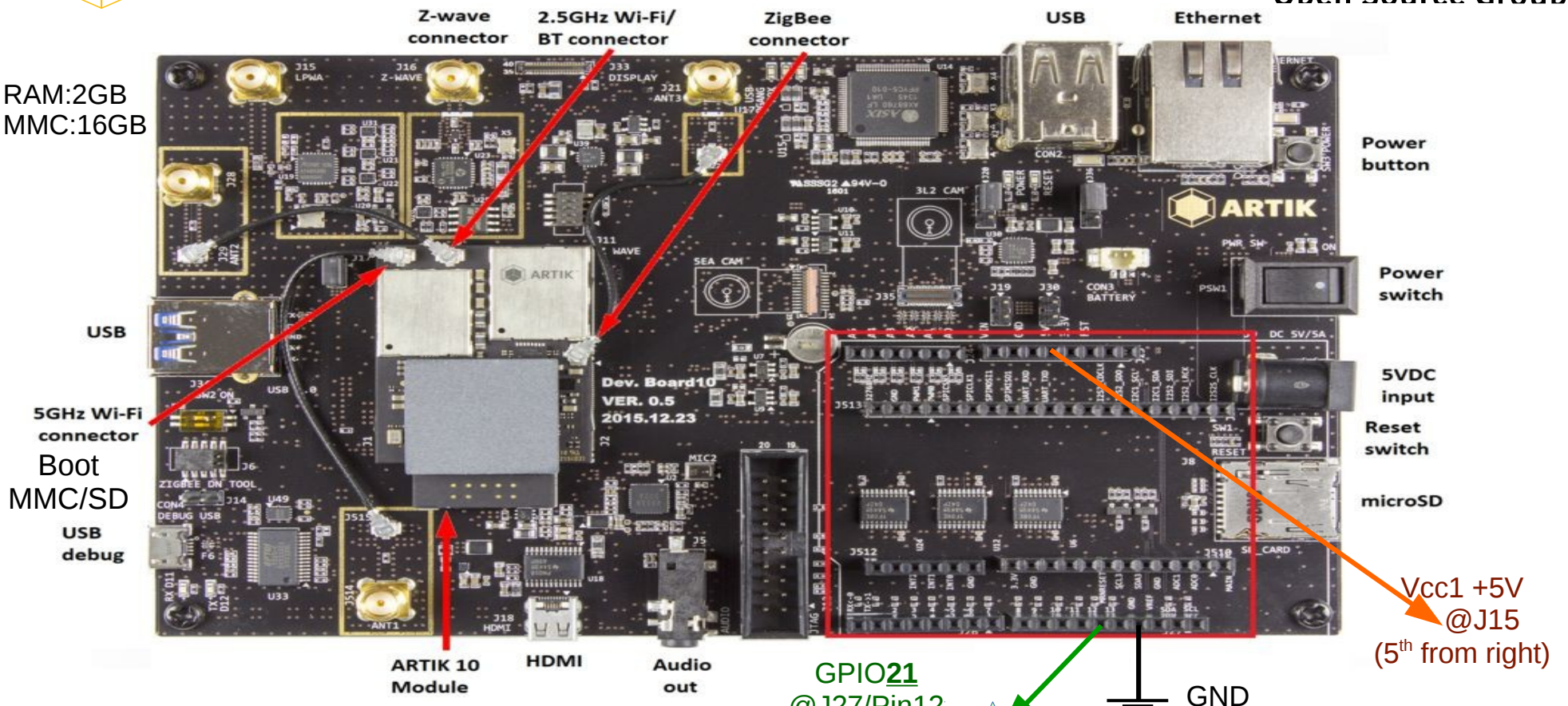
Resource is physical, not a boolean !

SAMSUNG
Open Source Group

- General Purpose ~~Input~~ Output: **GPIO**
  - Set a voltage on electrical pin from userspace
- This can be set using Linux's **sysfs** (adapt to C/C++)
  - echo $n > /sys/class/gpio/**export** ; echo out > /sys/class/gpio/gpio$n/**direction**
  - echo 1 ; sleep 1 ; echo 0 > /sys/class/gpio/gpio$gpio/**value**
- Or faster with direct access (kernel registers...)
  - Even better using mapping library RAA (Along UPM for sensors drivers)
- So, server's "**entity handler**" should send signal on **POST/PUT** requests, that's all
  - IoTServer::**handleEntity() { ...** IoTServer::handlePut() … }
  - IoTServer::handlePut() { ~ write("**/sys/class/gpio/gpio$n/value" ,**"%d", requestRep.**getValue**<bool>("*value*")); }

Samsung ARTIK™ 10 GPIO pinout

SAMSUNG
Open Source Group

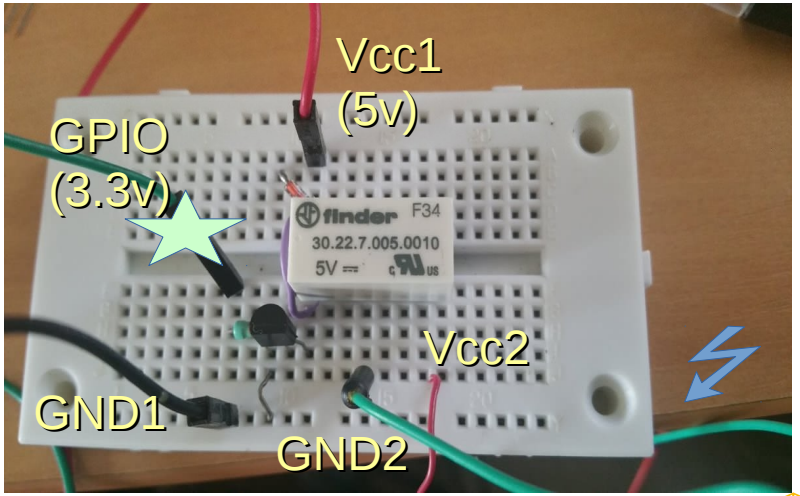


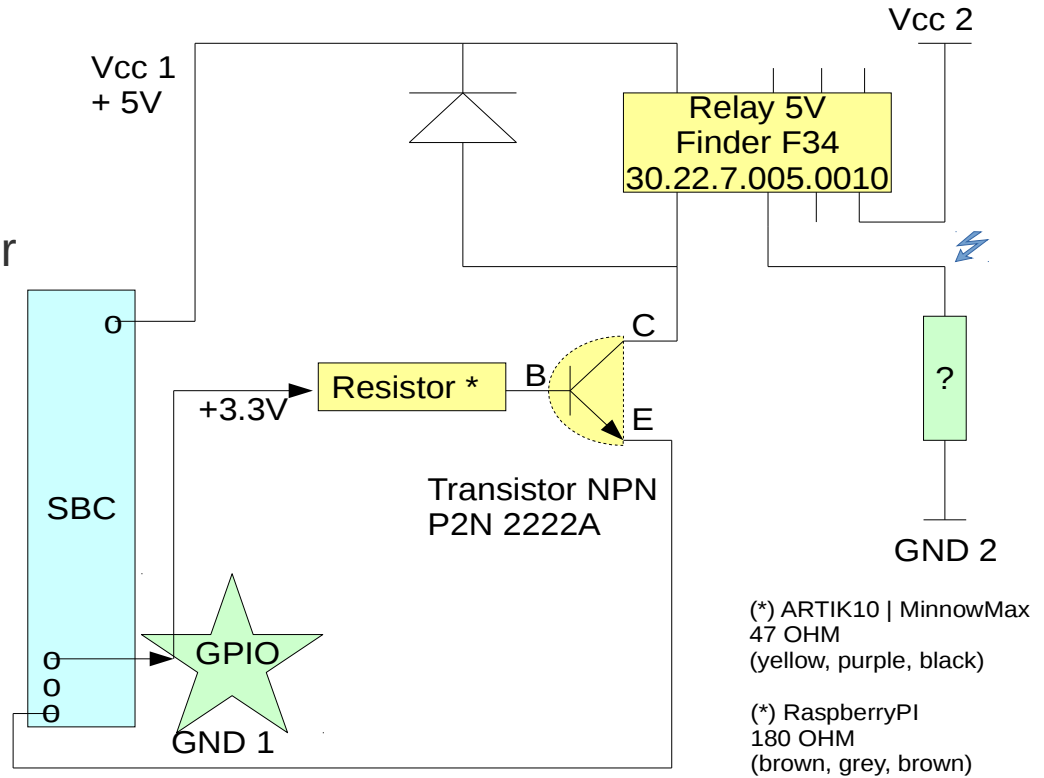
CPU: Exynos5 (4+4 Cores)

Hardware integration : DIY

SAMSUNG
Open Source Group

- High voltage relay (0-220V)
  - GPIO (3v3) < Relay (5V)
    - Signal Base of NPN Transistor



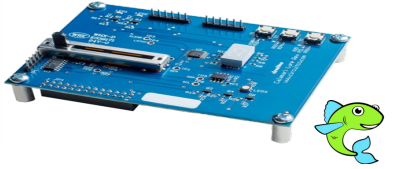


Samsung ARTIK™

Hardware integration, with modules

SAMSUNG
Open Source Group

- Simples modules, to wire on headers
  - Ie: Single channel Relay (HXJ-36)
- Daughters boards, (compatibles headers)
  - Shields: for Arduino, and compatibles SBC (ARTIK10, Atmel Xpl)
  - Hats for Raspberry Pi+ (RabbitMax ships relay, I2C, IR, LCD)
  - Lures for Minnowboard (Calamari has buttons, Tadpole transistors)
- Warning: Arduino Mega's GPIO is 5V and most SBC are 3.3V



IoT devices are constrained !

SAMSUNG
Open Source Group

- If GNU/Linux is not an option for the computing power you have
  - Now let's port it to MCU using C
  - CSDK : iotivity/resource/csdk
  - Can use the same code base for Linux | Arduino...
- Example:
  - git clone -b csdk iotivity-example
  - git clone -b arduino iotivity-example
  - AVR binary Footprint : 116534 bytes for ATMega2560

IoTivity CSDK flow

SAMSUNG
Open Source Group

IoTivity Server

IP Network

IoTivity Client(s)

```
OCInit(NULL, 0, OC_SERVER);
OCCreateResource( …, handleOCEntity);
        {  OCProcess();  }
```

```
OCInit(NULL, 0, OC_CLIENT);
OCDoResource(...,OC_REST_DISCOVER, ...)
 handleDiscover(... OCClientResponse ...)
```

```
handleOCEntity(entityHandlerRequest) {
  switch (entityHandlerRequest->method
{
    case 'POST:  // CREATE resource 1st

    case 'GET' : // READ current value

    case 'PUT' : // then UPDATE value
      ...
     OCDoResponse(&response);
 }}
```

```
OCDoResource(...OC_REST_POST …)
 handlePost(... OCClientResponse ...)
```

```
OCDoResource(...OC_REST_GET …)
 handleGet(... OCClientResponse ...)
```

```
OCDoResource(...OC_REST_PUT …)
 handlePut(... OCClientResponse ...)
```

Interaction with other OS / Devices

SAMSUNG
Open Source Group

- Consumer electronics products
  - Tizen ❤ IoTivity
    - Tizen:3 contains as platform package (.rpm)
    - Tizen:2 can ship lib into native app (.tpk)
      - For Samsung Z1 (Tizen:2.4:Mobile)
      - Samsung GearS2 (Tizen:2.3.1:Wearable)
- GNU/Linux:
  - Yocto (Poky, AGL, GENIVI, OstroOS)
- Other OS too:

TIZEN™

SAMSUNG
Open Source Group

"Any sufficiently
advanced technology
is indistinguishable
from **magic**."
~ *Arthur C. Clarke*

Demonstration: tizen-artik-20161010rzr

https://vimeo.com/186286428#tizen-artik-20161010rzr

SAMSUNG
Open Source Group

Want More ?

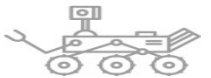
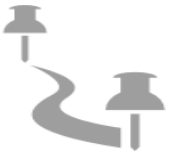
SAMSUNG
Open Source Group

- More security, enable it, device provisioning, iotcon...
- More constrained: iotivity-constrained (RIOT, Contiki, Zephyr)
- More connectivity: BT, BLE, Zigbee, LTE, NFC...
- Scale: Deploy an OCF network of sensors, establish rules.
  - Global: Webservices (WSI), with cloud backend
  - For Smart (Home | Car | City | $profile)

Conclusion

SAMSUNG
Open Source Group

- Prototyping an IoT device is possible
  - with IoTivity IoT framework, that provides
    - Device to Device seamless connection
    - Create, Read, Update, Delete Resource & Notification
      - Can be easly implemented In C or C++
  - On Single Board Computers supporting Linux
- To work with devices supporting OCF standard protocol
  - Or supporting IoTivity like Tizen Wearables
- Possibilities are infinites

References

SAMSUNG
Open Source Group

- Entry point:
  - https://wiki.iotivity.org/examples
- Technical references
  - https://openconnectivity.org/resources/iotivity
    - OIC_1.1_Candidate_Specification.zip
  - https://wiki.iotivity.org/sources
  - http://elinux.org/ARTIK
- Keep in touch online:
  - https://wiki.iotivity.org/community
  - https://wiki.tizen.org/wiki/Meeting
  - https://developer.artik.io/forums/users/rzr
  - https://blogs.s-osg.org/author/pcoval/

SAMSUNG
Open Source Group

Danke Schoen !
Thanks / Merci / 고맙습니다

Samung OSG, SSI,
Open Connectivity Foundation, LinuxFoundation,
FLOSS Communities: Tizen, Yocto, EFL, AGL, GENIVI
FlatIcons (CC BY 3.0) : Freepik, Chao@TelecomBretagne,
Libreoffice, openshot,
SRUK,SEF, Intel, Rabbitmax,
ELC/OpenIoT attendees,
YOU !

Contact:
https://wiki.tizen.org/wiki/User:Pcoval

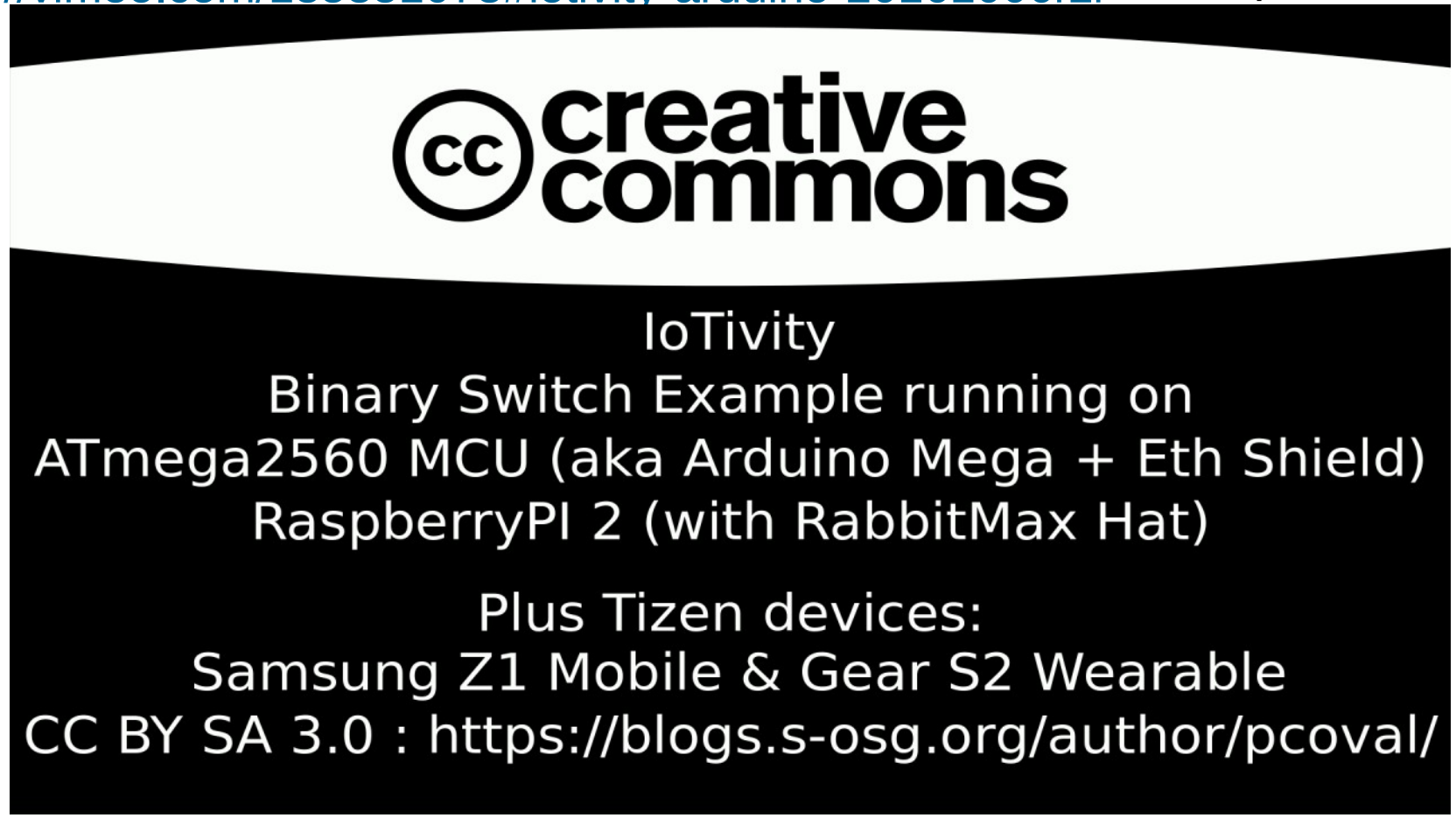
SAMSUNG
Open Source Group

Q&A or/and Annexes ?

Demonstration: iotivity-arduino-20161006rzr

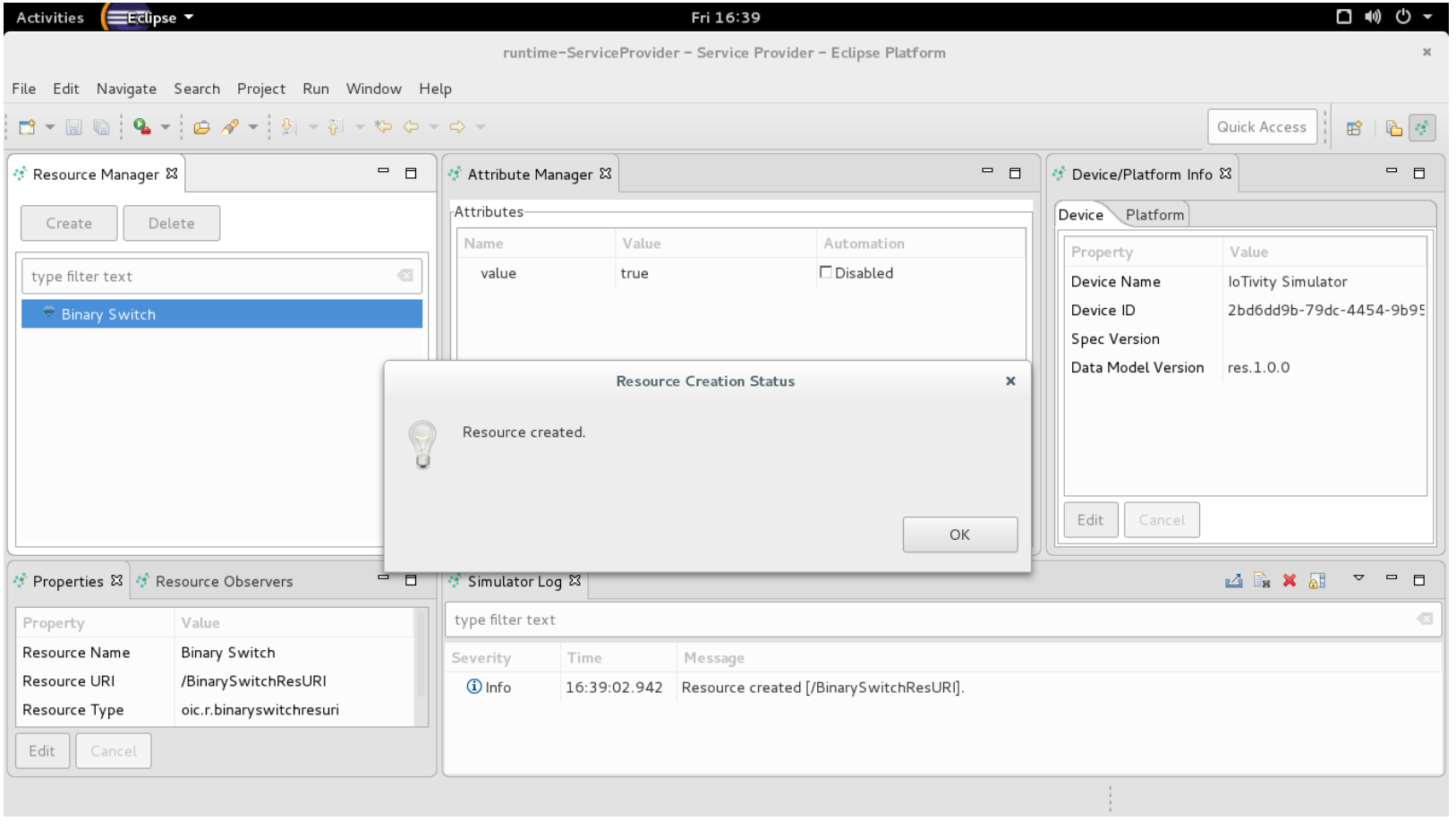
https://vimeo.com/185851073#iotivity-arduino-20161006rzr

SAMSUNG
Open Source Group



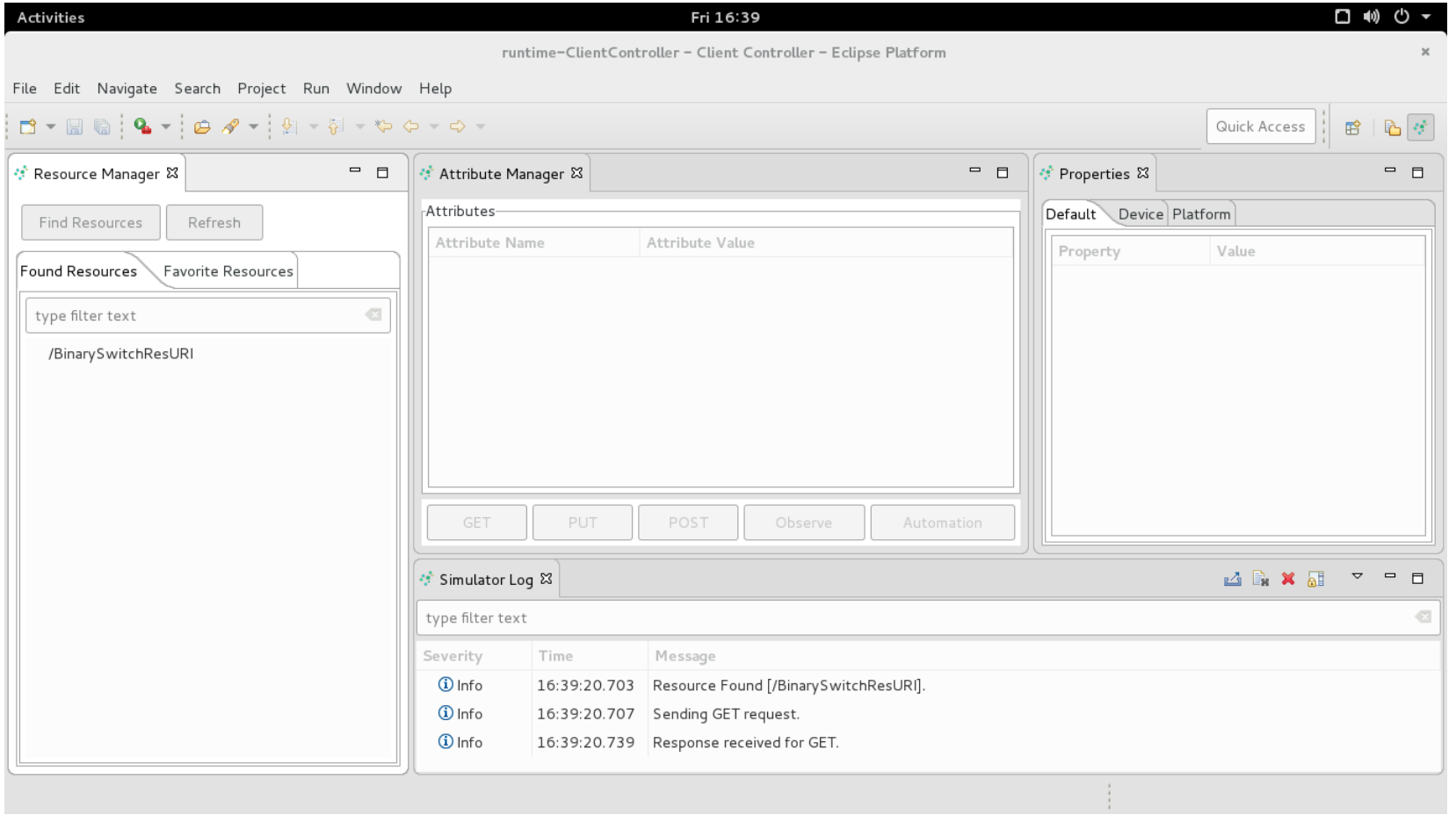
Simulator: Importing model, Resource served

SAMSUNG
Open Source Group



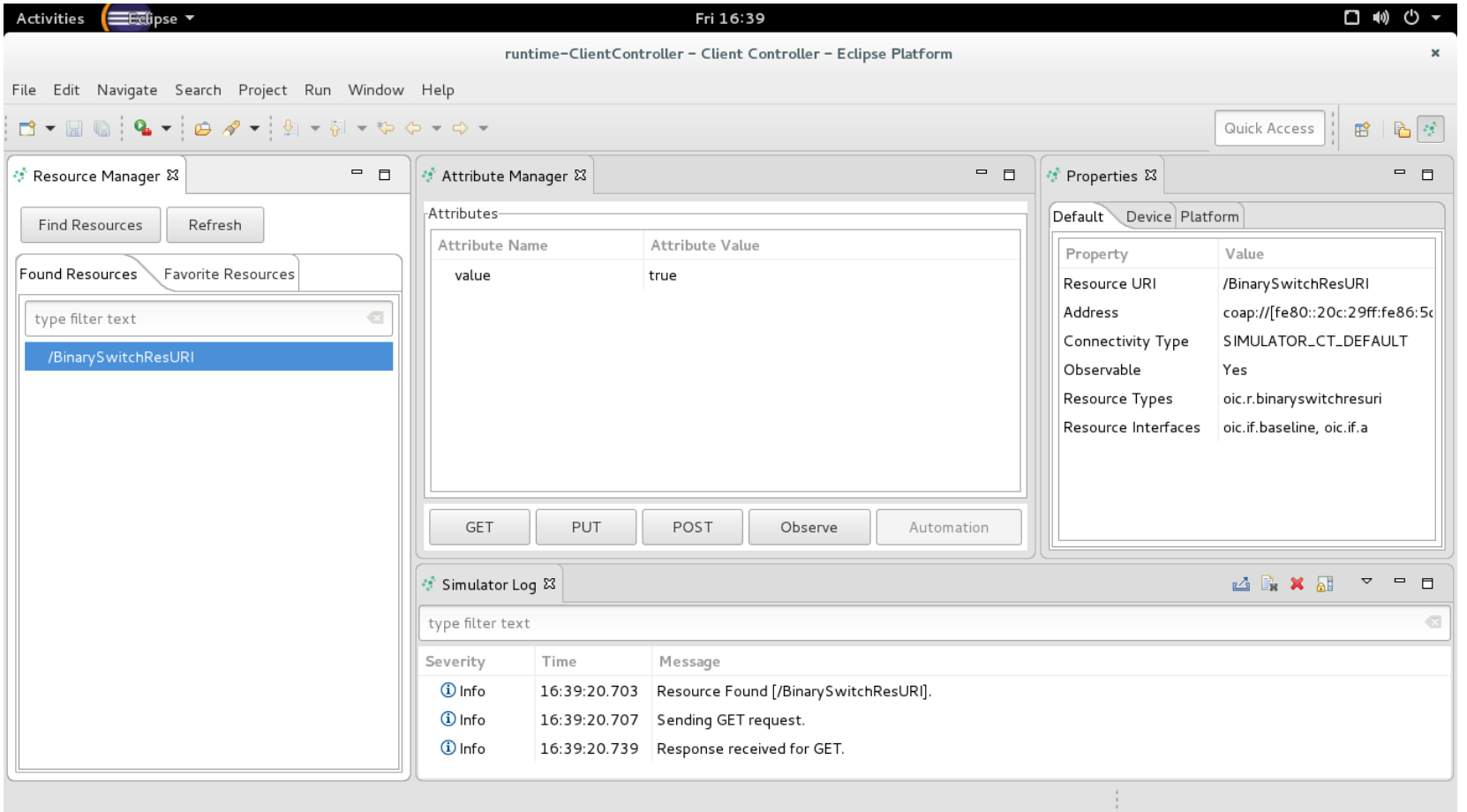
Client discovered Resource

SAMSUNG
Open Source Group



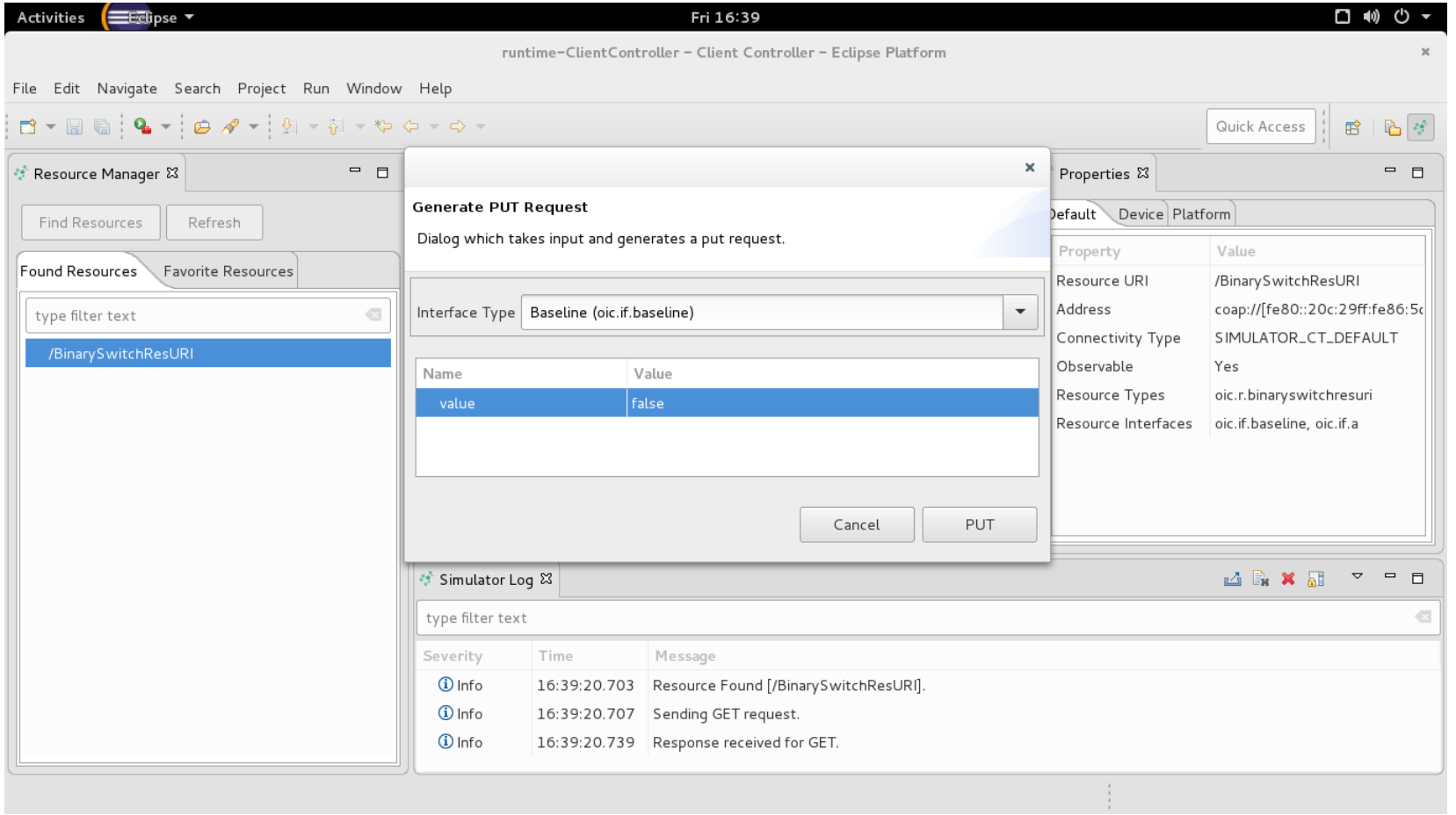
Resources properties on client

SAMSUNG
Open Source Group



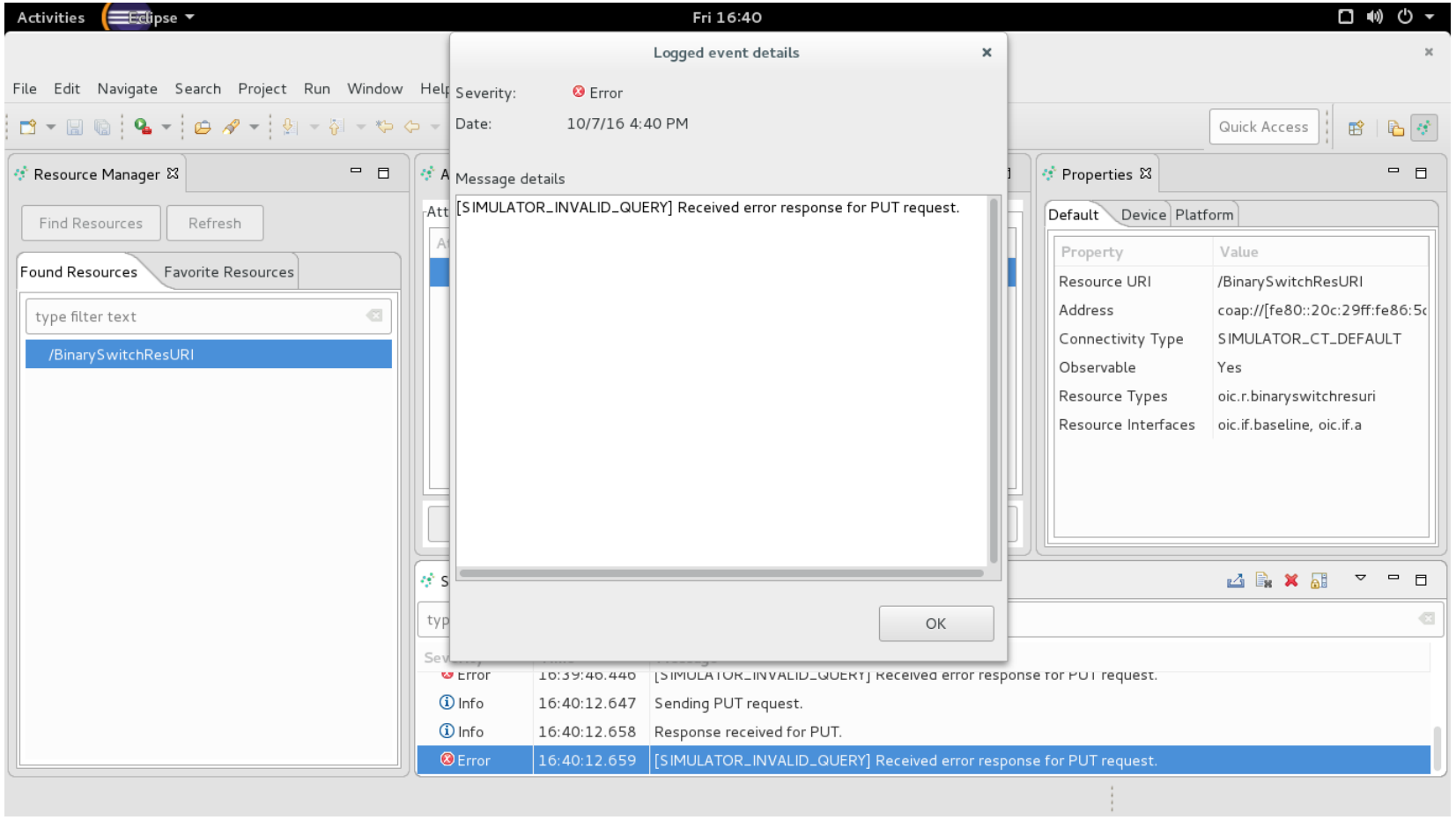
Attempt to change property using PUT

SAMSUNG
Open Source Group



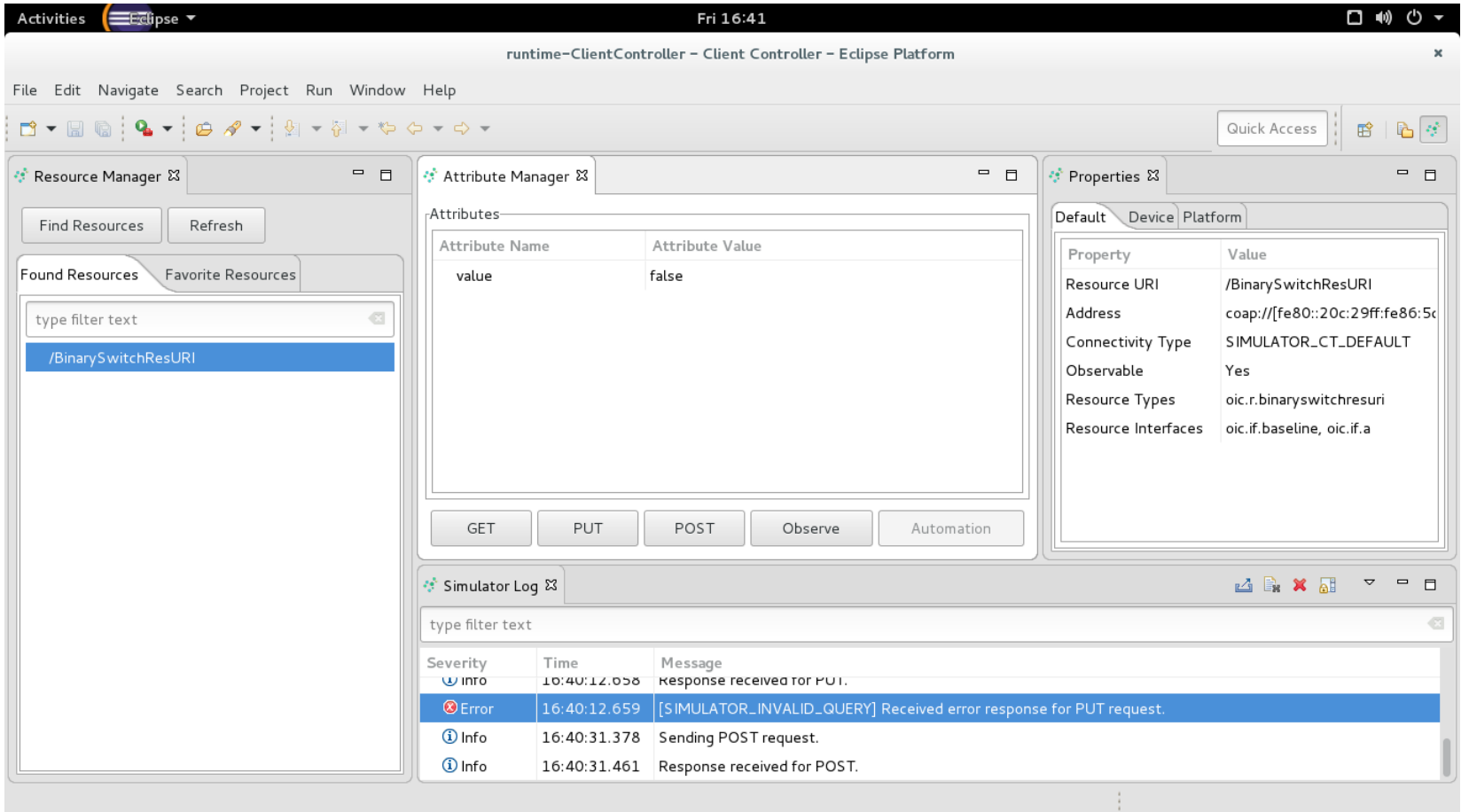
Failed, as unsupported by interface from model

SAMSUNG
Open Source Group



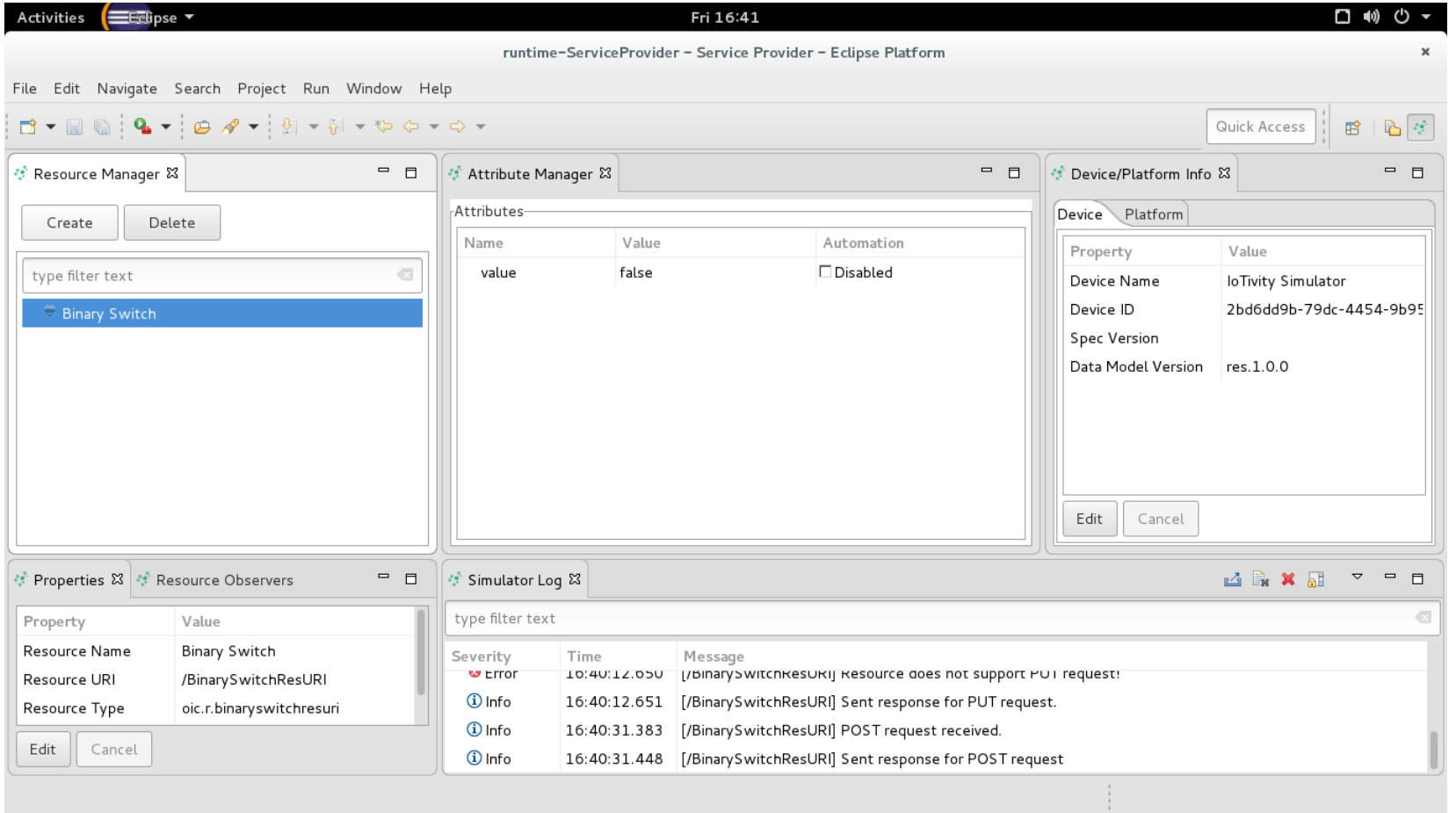
Client sets resource property using POST

SAMSUNG
Open Source Group



Server receives request and updates property

SAMSUNG
Open Source Group



Technical Showcase

CE Workgroup Linux Foundation / Embedded Linux Conference Europe

Tizen devices connected with IoTivity

Phil Coval / Samsung OSG

SAMSUNG Open Source Group

What is demonstrated

IoTivity

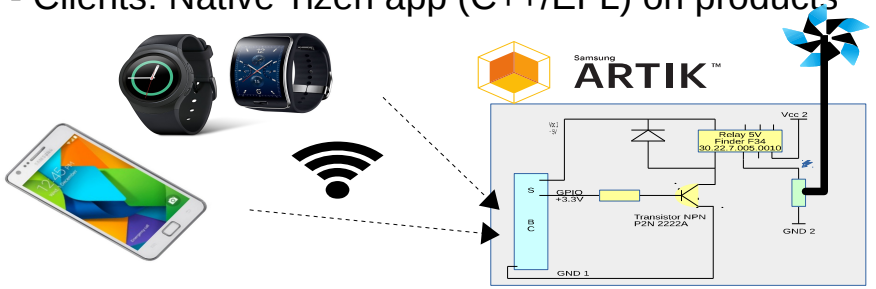
Seamless device to device connectivity framework

TIZEN

Linux-based software platform for consumer electronics

Demo: “Tizen DIY IoT Fan” controlled by devices

- Server: Tizen:Common on ARTIK dev board
- Clients: Native Tizen app (C++/EFL) on products



Hardware Information

ARTIK 10 (Exynos 5422 http://elinux.org/ARTIK)

Z1, Gear S (https://wiki.tizen.org/wiki/Devices)

What was improved

Tizen:Common 2016

- ARTIK 5 & 10 as latest reference devices
- Graphics: Enlightenment on Wayland

IoTivity 1.2.0

- Notification service
- Cloud features
- OS Support (Windows)
- CoAP (TSL, HTTP)
- UPnP bridge
- Extending support:
  - New OS: Windows, macOS
  - Linux: Tizen, Yocto, Debian, WRT...
  - Hardware: x86, ARM, RPi, MCU (Arduino)

Yocto project efforts (meta-oic, meta-artik...) yocto PROJECT

Source code or detail technical information availability

https://wiki.tizen.org/wiki/User:Pcoval

https://wiki.iotivity.org/community